Article
Brendan Bannon · Nov 29, 2016 7m read

The Art of Mapping Globals to Classes (3 of 3)

The Art of Mapping Globals to Classes (3 of 3)

If you are looking to breathe new life into an old MUMPS application follow these steps to map your globals to classes and expose all that beautiful data to Objects and SQL.

If the above does not sound familiar to you please start at the beginning with the following:

The Art of Mapping Globals 1

The Art of Mapping Globals 2

This example is going to show you how to map a classic parent-child structure.

Same disclaimer: If you can ' t make heads or tails of your globals after reviewing these articles please contact the WRC and we will try to help you out: Support@InterSystems.com.

Steps for Mapping a Global to a Class:

1. Identify a repeating pattern in the global data.
2. Identify what makes up a unique key.
3. Identify the properties and their types.
4. Define the properties in the class (don ' t forget the properties from the variable subscripts)
5. Define the IdKey index.
6. Define the Storage Definition:
    a. Define the Subscripts up to and including the IdKey.
    b. Define the Data section.
    c. Ignore the Row ID section. 99% of the time the default is what you want so let the system fill that in.
7. Compile and test your class / table.

In the last example each person could have 1 activity, but that would be pretty boring so we want to let people have more than one activity. In the example below step 1 gets a little more complex because there are 2 different sets of repeating data.

^ParentChild(1)="Brendan^45956"

^ParentChild(1,"Hobbies",1)="Pit Crew"

^ParentChild(1,"Hobbies",2)="Kayaking"

^ParentChild(1,"Hobbies",3)="Skiing"

^ParentChild(2)="Sharon^46647"

^ParentChild(2,"Hobbies",1)="Yoga"

^ParentChild(2,"Hobbies",2)="Scrap booking"

^ParentChild(3)="Kaitlin^56009"

^ParentChild(3,"Hobbies",1)="Lighting Design"

^ParentChild(3,"Hobbies",2)="pets"

^ParentChild(4)="Melissa^56894"

^ParentChild(4,"Hobbies",1)="Marching Band"

^ParentChild(4,"Hobbies",2)="Pep Band"

^ParentChild(4,"Hobbies",3)="Concert Band"

^ParentChild(5)="Robin^57079"

^ParentChild(5,"Hobbies",1)="Baking"

^ParentChild(5,"Hobbies",2)="Reading"

^ParentChild(6)="Kieran^58210"

^ParentChild(6,"Hobbies",1)="SUBA"

^ParentChild(6,"Hobbies",2)="Marching Band"

^ParentChild(6,"Hobbies",3)="Rock Climbing"

^ParentChild(6,"Hobbies",4)="Ice Climbing"

Step 1:

For this global there are 2 sets of repeating data, the first is at the first subscript level and has the personal info:

^ParentChild(1)="Brendan^45956"

^ParentChild(2)="Sharon^46647"

^ParentChild(3)="Kaitlin^56009"

^ParentChild(4)="Melissa^56894"

^ParentChild(5)="Robin^57079"

^ParentChild(6)="Kieran^58210"

The second is at the third subscript level and lists the hobbies:

^ParentChild(1,"Hobbies",1)="Pit Crew"

^ParentChild(1,"Hobbies",2)="Kayaking"

^ParentChild(1,"Hobbies",3)="Skiing"

^ParentChild(2,"Hobbies",1)="Yoga"

^ParentChild(2,"Hobbies",2)="Scrap booking"

…

So we are going to define 2 classes, one for each block of repeating data.

Step 2:

Two classes means we need to have 2 unique identifiers, one for each table: Example3Parent and Example3Child. This is easy for the parent table; it has only one subscript. For the child table it will be a compound key using the first subscript (the Parent Reference) and the third subscript (the childsub).

Step 3:

Looking at the data it is easy to identify 5 different properties. For Example3Parent we will have 3 properties: ParentId, Name, and DateOfBirth. For Example3Child we will have 2 properties: ChildId and Hobby. There are 2 more properties we need to define for this example. These are called Relationship Properties. There will be one in each class defining the relationship with the other class. A parent class can have many child classes, but a child class can have only one parent relationship.

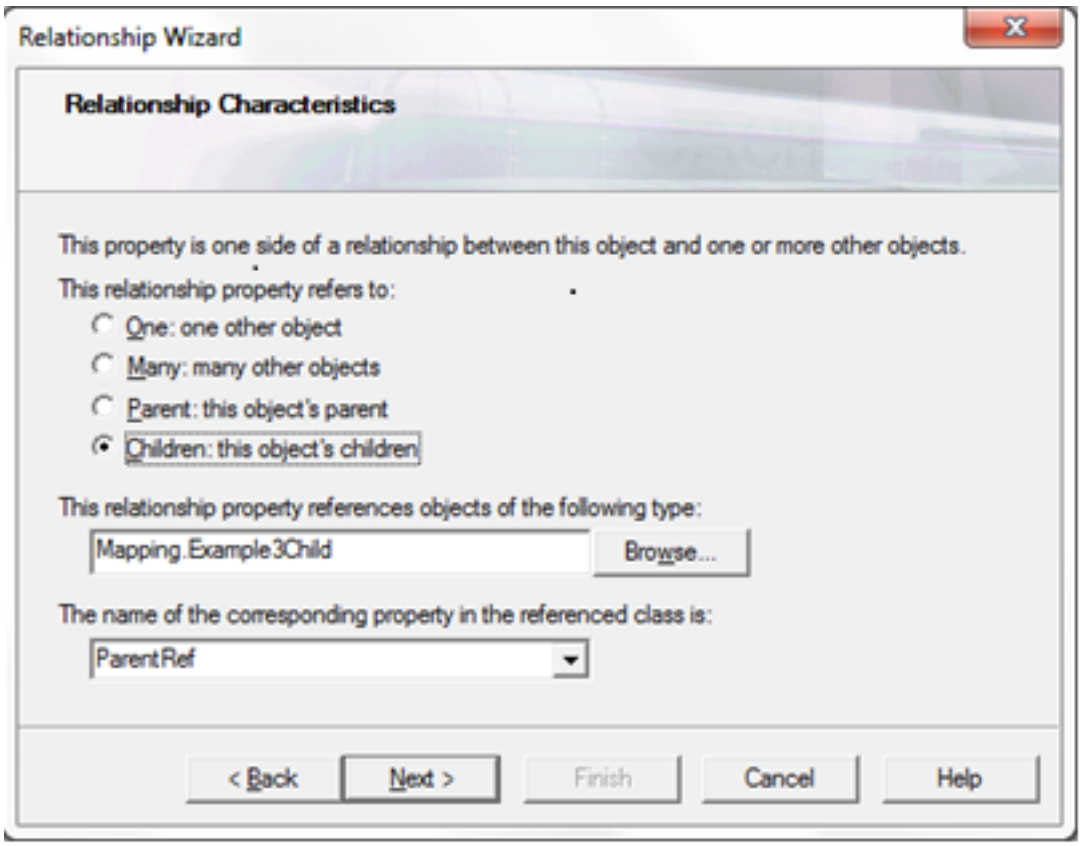
In the parent class we have:

Relationship HobbyRef As Mapping.Example3Child [Cardinality = children, Inverse = ParentRef];

In the child class we have:

Relationship ParentRef As Mapping.Example3Parent [Cardinality = parent, Inverse = HobbyRef];

You can define these properties by using the Property Wizard. If you click the Relationship radio button you get this page:



Step 4:

OK so I got a little ahead of myself in step three and showed you the Relationship Properties. Here are the others.

Example3Parent:

Property Name As %String;

Property DateOfBirth As %Date;

Property ParentId As %Integer;

Example3Child:

Property Hobby As %String;

Property ChildId As %Integer;

Step 5:

For Example3Parent the IdKey is easy, just subscript level 1:

```
Index Master On ParentId [ IdKey ];
```

For Example3Child the IdKey is based on subscript 1 and subscript 3 but for a parent-child relationship it can be a little tricky (if you have teenaged girls you understand). The IdKey index can list both the ParentRef and ChildId or it can just list the ChildId, both will compile correctly.

Index Master On (ParentRef, ChildId) [IdKey];

Or

Index Master On ChildId [IdKey];

Personally I think the first one should be the way it is done, but I am lazy so there is a good chance you will see the second in my examples.

Step 6:

There is nothing special about the mapping for Example3Parent so I am not going to go over that in detail.

Example3Child is a little different in the way you do the subscripts so I will go through the steps for that class.

Step 6a:

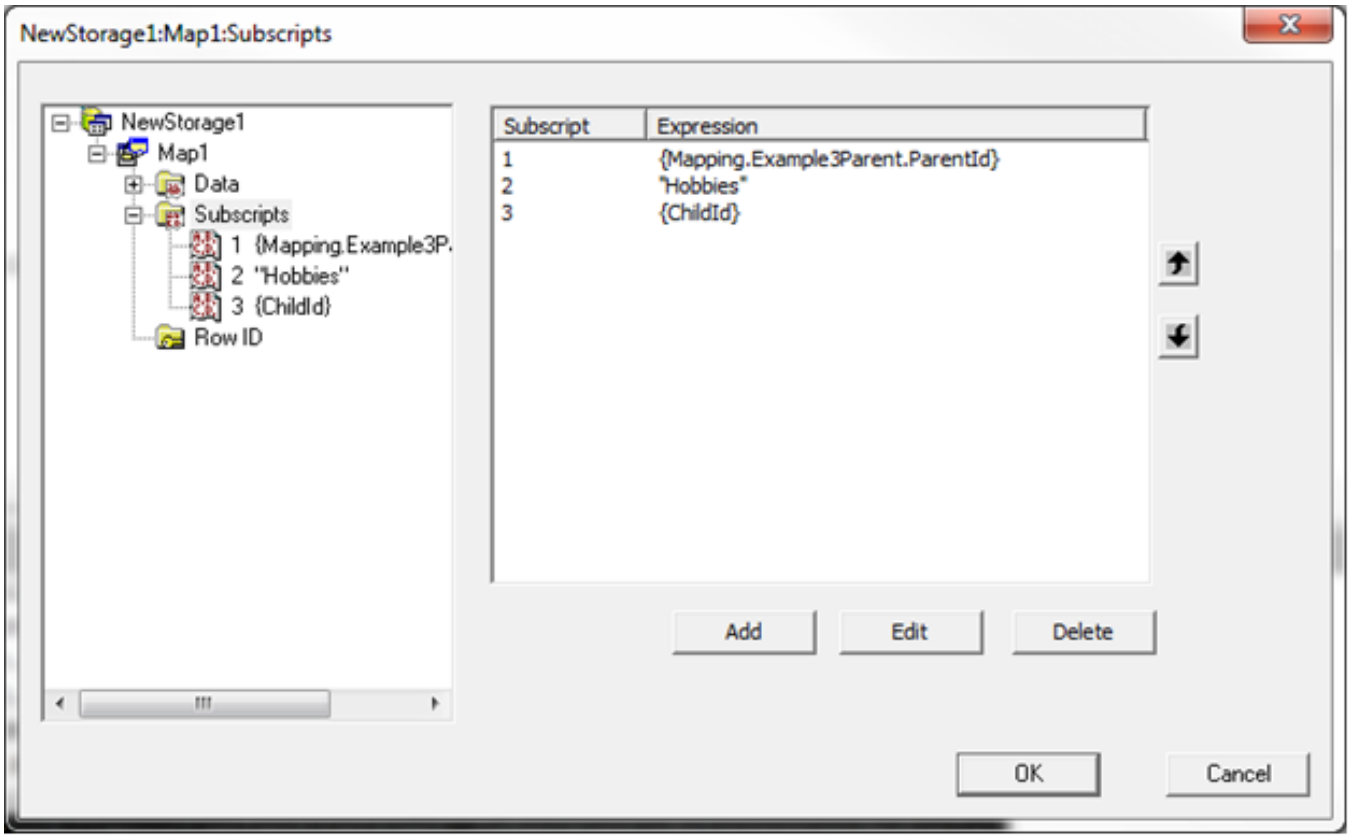
The global for the child class looks like this:

^ParentChild({ParentRef},"Hobbies",{ChildId})= ” ”

So we need 3 subscripts. The first one is the one that people have problems with. You don ’ t use the relationship property like I show above, instead you need to use the property from the parent class. To do this you use the full field syntax: {SchemaName.TableName.FieldName}. For this example the first subscript will be {Mapping.Example3Parent.ParentId}.

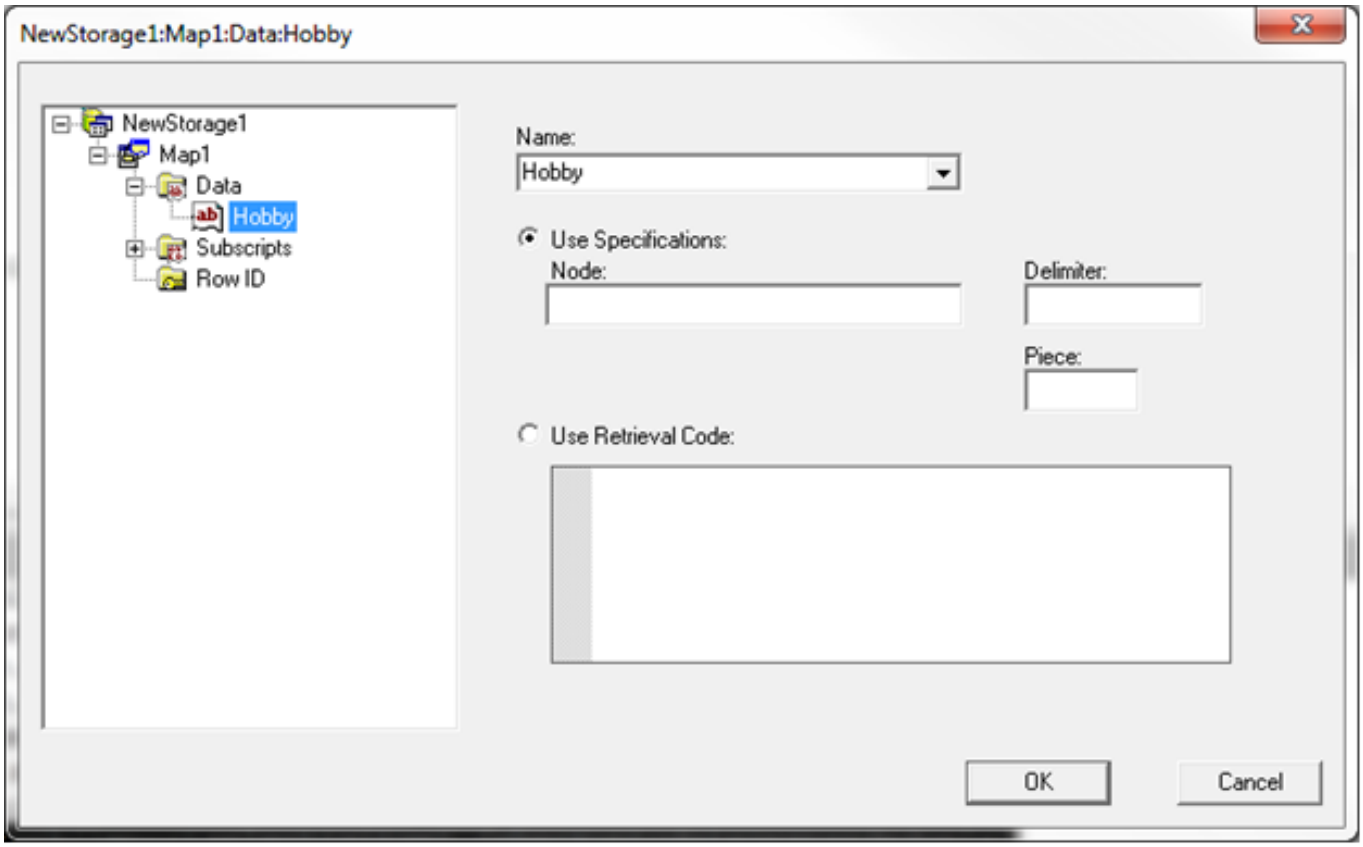
Subscript 2 is just the constant “ Hobbies ” .

Subscript 3 is the field {ChildId}.



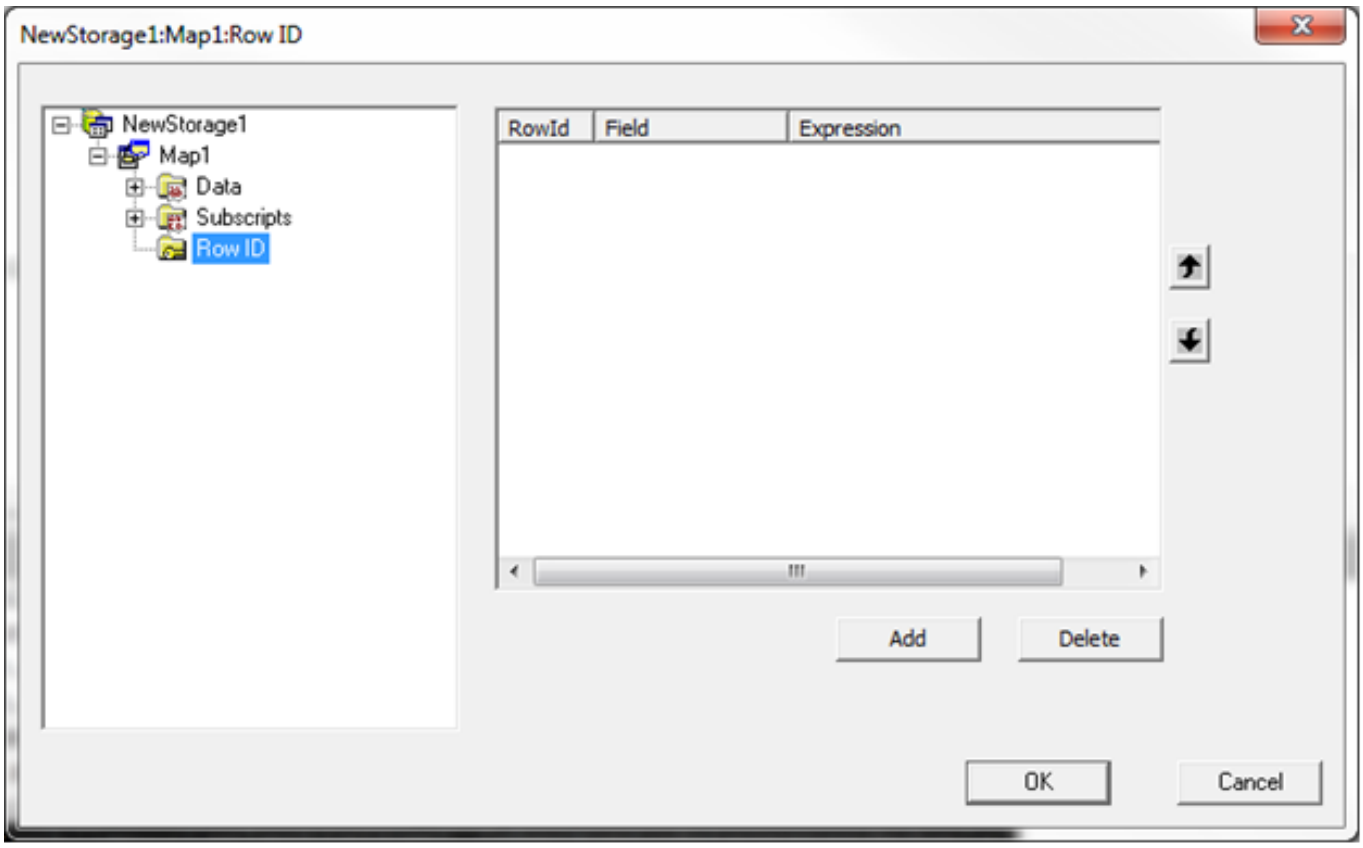
Step 6b:

The data section is super simple, just the field {Hobby}, no Piece and no Delimiter.



Step 6c:

All together now, “ Nothing to see here, just leave it blank ” .



Step 7:

All that is left is to do the compile:

```
Compilation started on 11/28/2016 08:26:31 with qualifiers 'fuk/importselectivity=1 /checkuptodate=expandedonly'
Compiling 2 classes, using 2 worker jobs
Compiling class Mapping.Example3Parent
Compiling class Mapping.Example3Child
Compiling table Mapping.Example3Parent
Compiling table Mapping.Example3Child
Compiling routine Mapping.Example3Parent.1
Compiling routine Mapping.Example3Child.1
Compilation finished successfully in 0.900s.
```

and a simple Join to make sure the data looks correct:

```
SELECT P.ParentId, P.Name, P.DateOfBirth, C.ID, C.Hobby

FROM Mapping.Example3Parent P

JOIN Mapping.Example3Child C ON P.ParentId = C.ParentRef
```

ParentId	Name	DateOfBirth	ID	Hobby
1	Brendan	10/28/1966	1¦¦1	Pit Crew

1	Brendan	10/28/1966	1||2	Kayaking
1	Brendan	10/28/1966	1||3	Skiing
2	Sharon	09/18/1968	2||1	Yoga
2	Sharon	09/18/1968	2||2	Scrap booking
3	Kaitlin	05/07/1994	3||1	Lighting Design
3	Kaitlin	05/07/1994	3||2	pets
4	Melissa	10/08/1996	4||1	Marching Band
4	Melissa	10/08/1996	4||2	Pep Band
4	Melissa	10/08/1996	4||3	Concert Band
5	Robin	04/11/1997	5||1	Baking
5	Robin	04/11/1997	5||2	Reading
6	Kieran	05/16/2000	6||1	SUBA
6	Kieran	05/16/2000	6||2	Marching Band
6	Kieran	05/16/2000	6||3	Rock Climbing
6	Kieran	05/16/2000	6||4	Ice Climbing

Note that there is a field called ID in the child class (there is a field / property called ID in every class) that is made up of the ParentId_" \|\| " ChildId.

For those that don't want to type here is a file with the globals and class: MappingExample3.zip

#Globals #Mapping #SQL #Caché